IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s):	Harm Sluiman		
Assignee:	International Business Machines Corporation		
Title:	Automation and Isolation of Software Component Testing		
Serial No.:	09/772,650	Filing Date:	January 30, 2001
Examiner:	Insun Kang	Group Art Unit:	2193
Docket No.:	CA920000042US1	Customer No.:	61136

Austin, Texas
January 2, 2008

Mail Stop Appeal Brief - Patents
Board of Patent Appeals and Interferences
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

SUPPLEMENTAL APPEAL BRIEF UNDER 37 CFR § 41.37

Dear Sir:

Applicant submits this Appeal Brief pursuant to the Notice of Appeal filed in this case on June 22, 2006. A Pre-Appeal Brief Request for Review and Statement of Reasons was filed with the Notice of Appeal. A Notice of Panel decision from the Pre-Appeal Brief review indicating to proceed to the Board of Patent Appeals and Interferences was mailed on September 8, 2006, thus setting the time period for filing an appeal to October 8, 2006 and pursuant to the Notification dated November 1, 2007. A Request for a One Month Extension is filed herewith setting a new time period to January 2, 2008 (January 1, 2008 being a Federal Holiday) The Appeal Brief has been previously paid electronically via the USPTO EFS. The one month extension fee is being paid electronically via the USPTO EFS. The Board is also authorized to deduct any other amounts required for this appeal brief and to credit any amounts overpaid to Deposit Account. No. 090461.

I. REAL PARTY IN INTEREST - 37 CFR § 41.37(c)(1)(i)

The real party in interest is the assignee, International Business Machines Corporation, as named in the caption above and as evidenced by the assignment set forth at Reel 011657, Frame 0536.

II. RELATED APPEALS AND INTERFERENCES - 37 CFR § 41.37(c)(1)(ii)

Based on information and belief, there are no appeals or interferences that could directly affect or be directly affected by or have a bearing on the decision by the Board of Patent Appeals and Interferences in the pending appeal.

III. STATUS OF CLAIMS - 37 CFR § 41.37(c)(1)(iii)

Claims 1 - 8 are pending in the application. Claims 1 - 8 stand rejected. The rejection of claims 1 - 8 is appealed. Appendix "A" contains the full set of pending claims.

IV. STATUS OF AMENDMENTS - 37 CFR § 41.37(c)(1)(iv)

No amendments after final have been requested or entered.

V. SUMMARY OF CLAIMED SUBJECT MATTER - 37 CFR § 41.37(c)(1)(v)

The present invention, as set forth by claim 1, relates to a method for testing software test components. (See e.g., Sluiman application, Page 5, lines 1 - 10.) The method includes ascertaining a public interface of the software test component (see e.g., Sluiman application, Page 7, lines 34 – 35) and creating a wrapper component for the software test component (see e.g., Sluiman application, Page 7, lines 35 - 36). Creating the wrapper component further includes defining a wrapper component interface to mirror the public interface of the test component and to receive calls to the software test component (see e.g., Sluiman application, Page 8, lines 2 - 3), defining the wrapper component to delegate to the software test component by including calls to the public interface of the software test component within the wrapper component (see e.g., Sluiman application, Page 8, lines 3 - 5), inserting test code within the wrapper component to permit capture and playback of user interaction with the public interface of the software test component (see e.g., Sluiman application, Page 10, lines 3 - 7), and enabling a test case to use the wrapper component interface to pass the received calls to the software test

component and to generate test data from the test code in the wrapper component (see e.g., Sluiman application, Page 11, lines 1 - 5).

The present invention, as set forth by claim 5 relates to a computer memory storing logic for testing software components (See e.g., Sluiman application, Page 5, lines 1 - 10). The logic includes first logic for ascertaining a public interface of a software test component, and second logic for creating a wrapper component for the software test component (see e.g., Sluiman application, Page 7, lines 35 - 36). The second code means includes logic for defining a wrapper component interface to mirror the public interface of the software test component(see e.g., Sluiman application, Page 8, lines 2 - 3), logic for defining the wrapper component to delegate to the software test component by including calls to the public interface of the software test component within the wrapper component (see e.g., Sluiman application, Page 8, lines 3 - 5), logic inserting test code within the wrapper component to permit capture and playback of user interaction with the public interface of the software test component (see e.g., Sluiman application, Page 8, lines 3 – 5), and logic for enabling a test case to use the wrapper component interface to access the software test component (see e.g., Sluiman application, Page 10, lines 3 - 7), and then to generate test data from the test code in the wrapper component (see e.g., Sluiman application, Page 11, lines 1 - 5).

The present invention, as set forth by claim 6 relates to a system in a computing environment for generation of a test environment for a software test component (See e.g., Sluiman application, Page 5, lines 1 - 10). The system includes means for ascertaining a public interface of the software test component, and means for creating a wrapper component for the software test component (see e.g., Sluiman application, Page 7, lines 35 - 36). The creating means includes means for defining a wrapper component interface to mirror the public interface of the software test component(see e.g., Sluiman application, Page 8, lines 2 - 3), means for defining the wrapper component to delegate to the software test component by including calls to the public interface of the software test component within the wrapper component (see e.g., Sluiman application, Page 8, lines 3 - 5), and means for making the wrapper component available to permit test code to be inserted within the wrapper component to permit capture and playback of user interaction with the public interface of the software test component (see e.g., Sluiman application, Page 8, lines 3 – 5), and to permit a test case component to use the wrapper

component interface to pass data to the software test component (see e.g., Sluiman application, Page 10, lines 3 - 7), and to generate test data from the test code in the wrapper component (see e.g., Sluiman application, Page 11, lines 1 - 5).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL - 37 CFR § 41.37(c)(1)(vi)

Claims 1 - 8 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Kobayashi, U.S. Patent No. 6,633,888 (Kobayashi) in view of Applicant's Admitted Prior Art (hereinafter APA).

VII. ARGUMENT - 37 CFR § 41.37(c)(1)(vii)

Claims 1- 8 Are Allowable Over Kobayashi, U.S. Patent No. 6,633,888 in view of Applicant's Admitted Prior Art

The present invention generally relates to providing a wrapper with an interface that mirrors an interface of a software test component. When a test case is applied to the wrapper, the wrapper interface is the same as would be the case for the software test component itself. The wrapper provides calls to the corresponding interface of the software test component and collects resulting data; but, is not a converter of signals for inconsistent interfaces.

Kobayashi generally relates to a method for creating and testing object oriented components with visual programming systems. More specifically, in Figure 2 of Kobayashi, the interface for test data applied to the proxy beans at box 216 is not the same as the interface of the universal transport API 206. For example, proxy beans have parameters represented by properties. It appears that box 206 translates the abnormal proxy bean data to the format recognized by normal class code at box 202, beans or that which is applied to the actual code at box 208. At col. 8, lines 23-27, Kobayashi teaches "each composite component in the application 216 can be tested within the visual builder by means of the universal transport API 206 which allows the code which implements the underlying objects and components 202 to be exercised under control of the proxy components." This does not disclose or suggest a test case interface mirrored on both sides of a wrapper. Kobayashi teaches adapters to allow a visual

editor to support editing of proxy beans and then testing of the edited code by control exercised through the universal transport API 206. The interface changes from box to box in Figure 2. This deficiency isn't overcome by the other prior art.

When responding to Applicant's arguments, the examiner has set forth:

> [T]he examiner points out again that a wrapper in the Java programming language is an object that encapsulates and delegates to another object for altering its behavior or interface. According to the application (specification, page 10), this wrapper operates as a "proxy" for the actual component and the "delegation code is generated for the wrapper's proxy classes to pass calls through to the component being tested." The generation of this wrapper through reflection to mirror the test component such as the limitations in the instant claim is possible through Java language features in the Java Bean specification. The instant specification also states that defining the wrapper component is possible by using tools such as the "introspection group of interfaces in the Java Bean specification." Such tools permit a wrapper generator to ascertain the members in the actual component to be used to define the proxy wrapper (page 10). Therefore, it is evident that the present invention simply uses the existing Java language features in the Java Bean specification to create a wrapper component, which acts as a proxy. That being said, Kobayashi's proxy bean acts as a delegate to the API of the actual bean and Kobayashi uses the parsing/extracting mechanism to determine/describe (i.e. introspections) and obtain (i.e. reflection) information about the members of a class such as the properties, methods, and constructs (i.e., "the parser/extractor 304 parses each constructor and each method and extracts any related fields, comments and parameter names," col. 8, lines 46-58). This extracting mechanism extends the conventional extraction process of "reflection" in the Java Bean specification so that the mechanism does not only determine the method parameters but also creates the "parameters to be converted to properties of the method bean created from the original method (col. 9 lines 1-19)." Using this extraction process (i.e. reflection), the APIs for all the classes can be retrieved and a proxy bean can be generated. The proxy component is to mirror the test component such as the wrapper in the instant claims. Therefore, Kobayashi discloses the limitation, defining a wrapper component interface to mirror the public interface of the software test component." Therefore, in view of the broadest reasonable interpretation, the rejection of the claims are considered proper and maintained. (Final office action dated March 23, 2006, pages 6, 7.)

Merely because the limitations of the claims are possible through features in the Java Bean specification does not mean that this same specification discloses or suggests such limitations. As discussed in the previous response, Applicant has found a clever way to tap into the data action during testing. The mirror interface allows the wrapper to support calls to access the interface of the software test component. Data is not changed to overcome an

incompatibility, it is collected for evaluation. These features are not disclosed or suggested by Kobayashi or the APA.

More specifically, Kobayashi and the APA do not disclose or suggest a method for testing software test component where the method includes creating a wrapper component for the software test component where creating the wrapper component further includes *defining a wrapper component interface to mirror the public interface of the test component and to receive calls to the software test component*, defining the wrapper component to delegate to the software test component by *including calls to the public interface of the software test component within the wrapper component, inserting test code within the wrapper component to permit capture and playback of user interaction with the public interface of the software test component*, and *enabling a test case to use the wrapper component interface to pass the received calls to the software test component and to generate test data from the test code n the wrapper component*, as required by claim 1. Claims 2 – 4 depend from claim 1 and are allowable for at least this reason. Claim 5 is a computer memory storing logic of similar scope to claim 1 and is allowable for similar reasons. Claim 6 is a system in a computing environment for generation of a test environment for a software test component of similar scope to claim 1 and is allowable for similar reasons. Claims 7 and 8 depend from claim 5 and are allowable for at least this reason.

VIII. CLAIMS APPENDIX - 37 CFR § 41.37(c)(1)(viii)

A copy of the pending claims involved in the appeal is attached as Appendix A.

IX. EVIDENCE APPENDIX - 37 CFR § 41.37(c)(1)(ix)

None

X. RELATED PROCEEDINGS APPENDIX - 37 CFR § 41.37(c)(1)(x)

There are no related proceedings.

XI. CONCLUSION

For the reasons set forth above, Applicant respectfully submits that the rejection of pending Claims 1 - 8 is unfounded, and requests that the rejection of claims 1 -8 be reversed.

I hereby certify that this correspondence is being electronically submitted to the COMMISSIONER FOR PATENTS via EFS on January 2, 2008.

/Stephen A. Terrile/

.....

Attorney for Applicant(s)

Respectfully submitted,

/Stephen A. Terrile/

Stephen A. Terrile
Attorney for Applicant(s)
Reg. No. 32,946

CLAIMS APPENDIX "A" - 37 CFR § 41.37(c)(1)(viii)

1. A method for testing a software test component, said method comprising the steps of:

ascertaining a public interface of the software test component; and

creating a wrapper component for the software test component by the substeps of:

defining a wrapper component interface to mirror the public interface of the software test component and to receive calls to the software test component, defining the wrapper component to delegate to the software test component by including calls to the public interface of the software test component within the wrapper component, inserting test code within the wrapper component to permit capture and playback of user interaction with the public interface of the software test component, and

enabling a test case to use the wrapper component interface to pass the received calls to the software test component and to generate test data from the test code in the wrapper component.

2. The method for testing a software test component according to Claim 1, wherein the software test component is an object-oriented software test component and wherein said step of ascertaining the public interface of the software test component further comprises interrogating a test component definition to determine public methods, constructor and associated parameters for the software test component.

3. The method for testing a software test component according to Claim 2, wherein the test component is a Java language class and said ascertaining step further comprises use of an introspection group of interfaces in a Java Bean specification.

4. The method for testing a software test component according to Claim 2, wherein the substep of defining a wrapper component interface further comprises defining public methods, constructors and associated parameters in the wrapper component to mirror the public methods, constructors and parameters determined for the software test component.

5. A computer memory storing logic for testing software components, the logic comprising:

first logic for ascertaining a public interface of a software test component; and

second logic for creating a wrapper component for the software test component, said second code means comprising:

logic for defining a wrapper component interface to mirror the public interface of the software test component,

logic for defining the wrapper component to delegate to the software test component by including calls to the public interface of the software test component within the wrapper component,

logic inserting test code within the wrapper component to permit capture and playback of user interaction with the public interface of the software test component, and

logic for enabling a test case to use the wrapper component interface to access the software test component and then to generate test data from the test code in the wrapper component.

6. A system in a computing environment for generation of a test environment for a software test component, said system comprising:

means for ascertaining a public interface of the software test component; and

means for creating a wrapper component for the software test component, said creating means including

means for defining a wrapper component interface to mirror the public interface of the software test component,

means for defining the wrapper component to delegate to the software test component by including calls to the public interface of the software test component within the wrapper component, and

means for making the wrapper component available to permit test code to be inserted within the wrapper component to permit capture and playback of user interaction with the public interface of the software test component and to permit a test case component to use the wrapper component

interface to pass data to the software test component and to generate test data from the test code in the wrapper component.

7. The system for generation of a test environment according to Claim 6, wherein the software test component is an object-oriented software test component and said means for ascertaining the public interface of the software test component further comprises

means for interrogating a definition of the software test component to determine public methods, constructor and associated parameters for the software test component.

8. The system for generation of a test environment according to Claim 7, wherein the means for defining a wrapper component interface comprises means for defining public methods, constructors and associated parameters in the wrapper component to mirror the public methods, constructors and parameters ascertained for the software test component.

EVIDENCE APPENDIX - 37 CFR § 41.37(c)(1)(ix)

None

RELATED PROCEEDINGS APPENDIX - 37 CFR § 41.37(c)(1)(x)

There are no related proceedings.